The layering method

Russell Impagliazzo

February 3, 2004

1 The general method

In the layering method for analyzing approximation algorithms, we first give a type of trivial instance, where any solution is approximately correct. We then show how to decompose the input into a "sum" of trivial instances, and similarly for solutions.

More precisely, we:

1. Define a class of "easy" instances
2. Prove that for any easy instance I, and any solutions S, S' for I, $Cost(S) \leq \rho Cost(S')$, so any solutions is within ρ of the optimal. (We could put in some "reasonableness" condition on S.)
3. Define a map from arbitrary instances I to a set of easy instances $E_1, ...E_m$. Intuitively, $I = E_1 + E_2 + ...E_m$, although this needn't be the case in any literal sense. We think of E_i as the i'th "layer" of I.
4. Define a map from solutions S of I to a sequence of solutions $S_1, ..S_m$ for $E_1, ..E_m$. Intuitively, $S_i = S \cap E_i$, that "part" of the solution from the i'th "layer" of the instance.
5. Prove that, for any solution S, $Cost(S, I) \geq \sum_{j=1}^{j=m} Cost(S_j, E_j)$, so the optimal solution for I has cost lower bounded by the sum of costs for the optimal solutions for each layer E_j.
6. Prove that, for the solution produced by our algorithm, S_A, $Cost(S^A, I) = \sum_{j=1}^{j=m} Cost(S_j^A, E_j)$, so the cost of our solution is bounded by the sum of costs of our solutions for each layer.
7. Conclude that our algorithm achieves approximation ratio ρ: Let S^{opt} be an optimal solution; from part 4, $Cost(S^{opt}, I) \geq \sum_{j=1}^{j=m} Cost(S_j^{opt}, E_j)$. From part 2, $\rho Cost(S_j^{opt}, E_j) \geq Cost(S_j^A, E_j)$, since any two solutions are within a factor ρ in cost for easy instances. Then $\rho Cost(S^{opt}, I) \geq \sum_{j=1}^{j=m} \rho Cost(S_j^{opt}, E_j). \geq \sum_{j=1}^{j=m} Cost(S_j^A, E_j) = Cost(S^A, I)$, (last step by Part 5) so our algorithm's solution is within ρ of optimal.

2 Minimum Cost Maximal Matching

Let the minimum cost maximal matching problem be as follows:

Instance: An undirected graph $G = (V, E)$ with positive edge weights $w(e) > 0$.

Solution format A set of edges $M \subseteq E$.

Constraints M is a matching, i.e., $\forall e_1, e_2 \in M$, $e_1 \neq e_2$, we have $e_1 \cap e_2 = \emptyset$ (no two edges share an endpoint) and M is maximal, i.e, $\forall e_1 \notin M$ there is an, $e_2 \in M, e_1 \cap e_2 \neq \emptyset$, i.e., adding any new edge to M violates the matching constraint.

Objective Minimize $Cost(M) = \sum_{e \in M} w(e)$.

2.1 The easy case

For minimum cost maximal matching, the easy case is the unweighted case, when all weights $w(e_i)$ are identical. Then the problem reduces to minimizing $|M|$, the number of edges used.

Lemma 1: Let M_1 and M_2 be maximimal matchings. Then $|M_1| \leq 2|M_2|$.

Proof: Let $e_1, ..e_k$ be the edges used in M_1. Let V_2 be the set of vertices that are endpoints of edges used in M_2, Each edge e_i must have some endpoint in V_2; otherwise, $M_2 \cup \{e_i\}$ would be a matching. Since the e_i are disjoint, then $|V_2| \geq k$. On the other hand, each edge in M_2 contributes two nodes in V_2, and they are disjoint, so $|M_1| \leq |V_2| = 2|M_2|$, as claimed.

Thus, pretty much any algorithm to achieves approximation ratio 2 on the unweighted cases.

2.2 The algorithm

We'll use a very simple greedy algorithm: Order the edges from smallest weight to largest weight. Consider the edges in order. Add each edge to M if it doesn't intersect with previously chosen edges.

Let $e_1, ...e_k$ be the edges chosen by this algorithm to get M. Let $\delta_1 = w(e_1)$, $\delta_i = w(e_i) - w(e_{i-1})$ for $2 \leq i \leq k$. Note that $\sum_{i=1}^{j} \delta_i = w(e_j)$.

2.3 The layering

We'll analyze the algorithm above using the layering method.

(I'm pretty sure some of the other methods we've gone through could be used instead, but it is a fairly direct and relatively simple illustration of layering, before we go on to the more complex vertex cover problem.)

Let E_i be the set of edges of weight $\geq w(e_i)$. Let G_i be the graph (V, E_i) where all edges have weight δ_i. This defines the layering of the instance.

Lemma 2: If $e \in E_i$, $w^G(e) \geq \sum_{j=1}^{j=i} w^{G_j}(e)$.

Proof: Since $e \in E_i$, $w(e) \geq w(e_i) = \sum_{j=1}^{j=i} \delta_j = \sum_{j=1}^{j=i} w^{G_j}(e)$.

Next, we'll define how to interpret an arbitrary solution as layered, and prove that the costs of the layers lower bounds the cost of the solution.

Lemma 3: If $overline M$ is a maximal matching, then $\overline{M}_i = \overline{M} \cap E_i$ is a maximal matching in E_i.

Proof: First, any subset of a matching is a matching. Second, if there were an edge $e' \in E_i$ so that $\overline{M} \cup \{e'\}$ were a matching in G_i, then $\overline{M} \cup \{e'\}$ would be a matching in G, and $\overline{M}$ wouldn't be maximal in G either.

Lemma 4: If $overline M$ is a maximal matching in G, and $\overline{M_i} = \overline{M} \cap E_i$, we have $Cost^G(\overline{M}) \geq \sum_{i=1}^{i=k} Cost^{G_i}(\overline{M_i})$.

Proof: $Cost^G(\overline{M}) = \sum_{e \in \overline{M}} w^G(e)$. Let $j(e)$ be the largest j so that $e \in E_j$. By Lemma 2, for each edge $e \in \overline{M}, w^G(e) \geq \sum_{j=1}^{j=j(e)} w^{G_j}(e) = \sum_{i, e \in E_i} w^{G_i}(e)$. Hence, $Cost^G(\overline{M}) \geq \sum_{e \in \overline{M}} \sum_{i, e \in E_i} w^{G_i}(e) = \sum_{i=1}^{i=k} \sum_{e \in \overline{M}_i} w^{G_i}(e) = \sum_i Cost^{G_i}(\overline{M}_i)$.

Lemma 5: For our algorithm's matching M, $Cost^G(M) = \sum_i Cost^{G_i}(M_i)$.

Proof: Note that for each $e_i \in M$, $e_i \in E_1, ...E_i$ and $w(e_i) = \sum_{j \leq i} \delta_j$. So $\sum_i Cost^{G_i}(M_i) = \sum_i \sum_{j \geq i}(\delta_i) = \sum_i \sum_{j \leq i}(\delta_j) = \sum_i w(e_i) = Cost^G(M)$.

2.4 Conclusion

Theorem: The greedy algorithm produces a maximal matching at most twice the optimal cost.

Proof: Let M be the maximal matching produced by our algorithm, and let $\overline{M}$ be a minimum cost maximal matching. By Lemma 5, $Cost^G(M) = \sum_i Cost^{G_i}(M_i) = \sum_i \delta_i |M_i|$. By Lemma 3, each M_i and $\overline{M}_i$ is a maximal matching in G_i, so by Lemma 1, $|M_i| \leq 2|\overline{M}_i|$. Thus, $Cost^G(M) \leq 2 \sum_i \delta_i |\overline{M}_i| = 2 \sum_i Cost^{G_i}(\overline{M}_i) \leq 2 Cost^G(\overline{M})$ by Lemma 4. Thus, M is within a factor 2 of the optimal cost.

3 Minimum Weight Vertex Cover

Let the minimum weight vertex cover problem be as follows:

Instance: An undirected graph $G = (V, E)$ with positive node weights $w(v) > 0$.

Solution format A set of nodes $C \subseteq V$.

Constraints C is a vertex cover, i.e., $\forall e \in E$, $e \cap C \neq \emptyset$.

Objective Minimize $Cost(C) = \sum_{v \in C} w(v)$.

3.1 The easy case

We saw an easy unweighted case that reduced to minimum cost maximal matching, but the easy case for layering is somewhat different. The easy case is the degree proportional case, $w(v) = \alpha d(v)$, where $\alpha > 0$ and $d(v)$ is the degree of vertex v in G. Intuitively, in this case, all v give us the same "coverage per dollar", covering $d(v)$ edges for $\alpha d(v)$ cost. So "it doesn't matter much" which cover we pick.

We formalize this as follows:

Lemma 1: Let G be a weighted graph, where $w(v) = \alpha d(v)$ for some $\alpha > 0$ and each $v \in V$. Let $m = |E|$ be the number of edges. Let C be a vertex cover. Then $\alpha m \leq Cost^G(C) \leq 2\alpha m$.

Proof: First, $Cost^G(C) = \sum_{v \in C} w(v) = \alpha \sum_{v \in C} d(v) = \alpha \sum_{v \in C} \sum_{e | v \in e} 1 = \alpha \sum_e \sum_{v \in C \cap e} 1 = \alpha \sum_e |e \cap C| \geq \alpha m$, since all edges have at least one end-point in C. On the other hand, $Cost^G(C) = \sum_{v \in C} w(v) = \alpha \sum_{v \in C} d(v) \leq \alpha \sum_{v \in V} d(v) = 2\alpha m$, since the sum of degrees of all nodes in a graph is twice the number of edges.

Corollary 2: Let G be a weighted graph, where $w(v) = \alpha d(v)$ for some $\alpha > 0$ and each $v \in V$. Let C_1, C_2 be vertex covers. Then $Cost^G(C_1) \leq 2Cost^G(C_2)$.

A second easy case is a graph with no edges. Clearly, the mimimum weight vertex cover in such a graph is the empty set.

3.2 The decomposition and the Johnson algorithm

How can we write an arbitrary graph as a "sum" of these degree-weighted graphs ? (Actually, our decomposition will also have one graph with no edges.)

Assume G has at least one edge. Let $\alpha_1 = min_v w(v)/d(v)$. Then if we subtract weight $\alpha_1 d(v)$ from every $w(v)$, we never get a negative number, and at least one weight becomes 0. So let G_1 be the graph with all nodes and edges from G, and with node weights $w_1(v) = \alpha_1 d_{G_1}(v)$. Then let the residual graph R_2 be the graph $V_2 = \{v \in V | w_1(v) < w(v)\}$, with the induced set of edges, and the residual weights $\overline{w}_2(v) = w(v) - w_1(v)$. By construction, R_2 has only postive weights, and it has at least one fewer node than G.

We iterate this process. Say that we've constructed $G_1, ..G_{i-1}$ and $R_i = (V_i, E_i, \overline{w}_i)$. If R_i has no edges, we stop. Otherwise, we compute $\alpha_i = min_v \overline{w}_i(v)/d_{R_i}(v)$. We let G_i be the graph with the same nodes and edges as R_i, but with weights $w_i(v) = \alpha_i d_{R_i}(v) = \alpha_i d_{G_i}(v)$ (since G_i has the same edges as R_i). We then let $V_{i+1} = \{v \in V_i | \overline{w}_i(v) > w_i(v)\}$, E_{i+1} be the induced set of edges, and let $\overline{w}_{i+1}(v) = \overline{w}_i(v) - w_i(v)$ for each $v \in V_{i+1}$.

Note that this construction can be performed in polynomial time. For each iteration involves taking the minimum of n quantities that can be computed from R_i in time $O(n + m)$, and then performing $O(n)$ operations to get the new weights. Also, $|V_i|$ shrinks each iteration, so the number of iterations is at most n. So we can compute the entire sequence of graphs $G_1, ..G_t, R_{t+1}$ in time

$O(n^2+nm)$. Our algorithm will be to add the nodes v of the mimimum possible ratio $\alpha_{i+1} = \overline{w}_i(v)/d_{R_i}(v)$ to C. Equivalently, since those nodes are $V_{i+1} - V_i$, we eventually add all nodes except those in V_{t+1} to C, so $C = V - V_{t+1}$. This algorithm was invented by David Johnson.

(So here, unlike the maximal matching problem, the layering gives us our algorithm, not just its analysis.)

Note that each G_i is the "easy" case from the first subsection, where each node has weight proportional to its degree. The remaining residual graph R_{t+1} has no edges, so it is also an easy case.

3.3 Decomposing solutions

Let C be any vertex cover. Let $C_i = C \cap V_i$ for $i = 1..t+1$.

Lemma 3: If C is a vertex cover in G, then for each $1 \le i \le t$, C_i is a vertex cover in G_i.

Proof: If not, there is an edge $e \in E_i$ so that neither endpoint is in $C_i = C \cap V_i$. Since C is a vertex cover of G, and $E_i \subseteq E$, e must have an endpoint $u \in C$. But since $e \in E_i$, both endpoints are in V_i, so $u \in C \cap V_i$, a contradiction.

Lemma 4: For any vertex cover C, $Cost^G(C) = \sum_{i=1}^{i=t} Cost^{G_i}(C_i) + Cost^{R_{t+1}}(C_{t+1})$.

Proof: $Cost^G(C) = \sum_{u \in C} w(u) = \sum_{u \in C} \sum_{1 \le i \le t | u \in V_i} w_i(u) + \sum_{u \in C \cap V_{t+1}} \overline{w}_{t+1}(u) = \sum_{1 \le i \le t} \sum_{u \in C \cap V_i} w_i(u) + Cost^{R_{t+1}}(C_{t+1}) = \sum_{1 \le i \le t} Cost^{G_i}(C_i) + Cost^{R_{t+1}}(C_{t+1}) == \sum_{1 \le i \le t} Cost^{G_i}(C_i) + Cost^{R_{t+1}}(C_{t+1})$

Corollary 5: For any vertex cover C, $Cost^G(C) \ge \sum_{i=1}^{i=t} Cost^{G_i}(C_i)$

Proof: $Cost^{R_{t+1}}(C_{t+1}) \ge 0$

Corollary 6: For the Johnson's algorithm's vertex cover, C^J, $Cost^G(C^J) = \sum_{i=1}^{i=t} Cost^{G_i}(C_i^J)$

Proof: By construction, $C_{t+1}^J = \emptyset$, so $Cost^{R_{t+1}}(C_{t+1}) = 0$

3.4 Conclusion

Theorem: Johnson's algorithm produces a vertex cover of at most twice the optimal cost.

Proof: Let C^J be the vertex cover of the Johnson's algorithm, and let C be an optimal vertex cover. By Corollary 6, $Cost^G(C^J) = \sum_{i=1}^{i=t} Cost^{G_i}(C_i^J)$. Since by Lemma 3, each C_i^J and C_i are vertex covers of G_i, and G_i is an easy case, by Corollary 2 we have $Cost^{G_i}(C_i^J) \le 2Cost^{G_i}(C_i)$. Thus, $Cost^G(C^J) = \sum_{i=1}^{i=t} Cost^{G_i}(C_i^J) \le 2\sum_{i=1}^{i=t} Cost^{G_i}(C_i) \le 2Cost^{G_i}(C)$ by Corollary 5.